

## Q55 : 跳跃游戏 nums[i]:表示可以跳的最远距离。

1. 动态规划:  $f(i)$ :表示位置*i*是否可达。  $f(i) = E(\text{存在}j\text{使表达式为true}) : f(j < i) == \text{true} \ \&\& \ j + \text{nums}[j] \geq i$ .  
 $O(n^2)$
2. 贪心: 我们在每一个位置, 总是可以直到当前位置可以到达的最远位置。那么我们只需要保存最远位置即可。 $O(n)$  我们在最远位置之内都可以到达, 到达一个位置, 就去更新, 可到达的最远位置。

## Q621 : 任务调度器 tasks = ["A","A","A","B","B","B"], n = 2 还是比较复杂的相同的任务要间隔 2秒。每个任务要1秒完成。

```
tasks = [A,A,A,A,B,B,B,B,C,C,C,D,D,E,E] n = 2 len = tasks.len
```

```
A B C D E    5
A B C D      4
A B C E      4
A B          2
```

共 15秒。中间没有等待时间。 等待时间在哪里产生?

```
tasks = ["A","A","A","B","B","B"], n = 2
A B _
A B _
A B
```

1. 假设A的次数为k+1, 等待区在 第一列A 右侧 的  $n*k$ 区域。如果这个区域不能被填满, 那么就会产生等待区。
2. 由任务最多的A来生成等待区。
3. 填满等待区后, 右侧的可以从上到下, 一直填。并且 **不会产生等待区**。
4. 我们初始化一个 值  $quyu = k*n$  我们使用除了A的其他元素来占用等待区。
5. 如果等待区可以被占满, 那么 总时间就是 数组长度, len
6. 否则就是 len + 空闲的等待区quyu。
7. 将任务分为 3种。 A属于次数最多的一种。 B属于和A次数一样的一种。 CDE的次数都是  $\leq A-1$ 的 我们总是把 A放在第一列。

```
for (char c: tasks)
    map[c - 'A']++;
Arrays.sort(map);
int max_val = map[25] - 1, idle_slots = max_val * n;
for (int i = 24; i >= 0 && map[i] > 0; i--) {
    idle_slots -= Math.min(map[i], max_val);
    // 若次数和第一列相同, 也只能占用 max_val的等待区。
}
```

```
        return idle_slots > 0 ? idle_slots + tasks.length : tasks.length;
    }
}
```

**Q763** : 划分字母区间 将字符串划分为 尽可能多 的字符串。 同一个字符不能出现在多个子串中。

1.  $S = \text{"ababcbacadefegdehijhklij"}$  划分结果为  $\text{"ababcbaca", "defegde", "hijhklij"}$ 。三个子串的字符种类都不同。
2. 使用last数组，记录字符最后出现的位置。
3. 对于  $\text{last}[s[0] - 'a'] = \text{end1}$ 。那么第一个串最早于end1结束。  $\text{last}[s[1] - 'a'] = \text{end2}$  若  $\text{end2} > \text{end1}$  那么第一个串最早于end2结束。直到  $i = \text{end}$ 时，第一个串划分结束。

**Q767** : 重构字符串 给定一个字符串，要求按照相同元素不相邻的原则，重新排列。如果可以重新做到，返回该串，否则返回空串。

1. 使用优先队列。队列中放 字符。字符按照字符频率进行排序。每次选出两个最多的字符追加到ans. 直到所有字符被用光。如果一个字符被用光，则从队列中删除

```
PriorityQueue<Character> queue = new PriorityQueue<Character>(new
Comparator<Character>() {
    public int compare(Character letter1, Character letter2) {
        return counts[letter2 - 'a'] - counts[letter1 - 'a'];
    }
});
```

2. 计数的方式，如果一个字符的频率为  $(n+1)/2$  时，那么该字符一定要占据所有偶数位置。否则一定会相邻。我们维护 oddind奇数下标，和evenind偶数下标。使用for(0:26)遍历a到z.对所有的字符 x, 若字符x的频率小于 $(n+1)/2$ 则优先使用奇数oddind位置。当奇数位置被占用完了。再往偶数even位置安排。